

```

unsigned x[10], y[10], i, j, k, dir;

void exch (unsigned pos1, unsigned pos2) {
    unsigned tmp;
    tmp=y[pos1]; y[pos1]=y[pos2]; y[pos2]=tmp;
}
unsigned need_exch (unsigned pos1, unsigned pos2) {
    if (dir==0) {
        return (y[pos1]<y[pos2]);
    }
    else {
        return (y[pos1]>y[pos2]);
    }
}

void main () {
    for (i=0;i<10;i=i+1) y[i]=x[i];
    for (i=9;i>=1;i=i-1) {
        for (j=0; j<i; j=k) {
            k = j+1;
            if (need_exch(j, k))
                exch(j, k);
        }
    }
}

```

***FIG. 1***

```

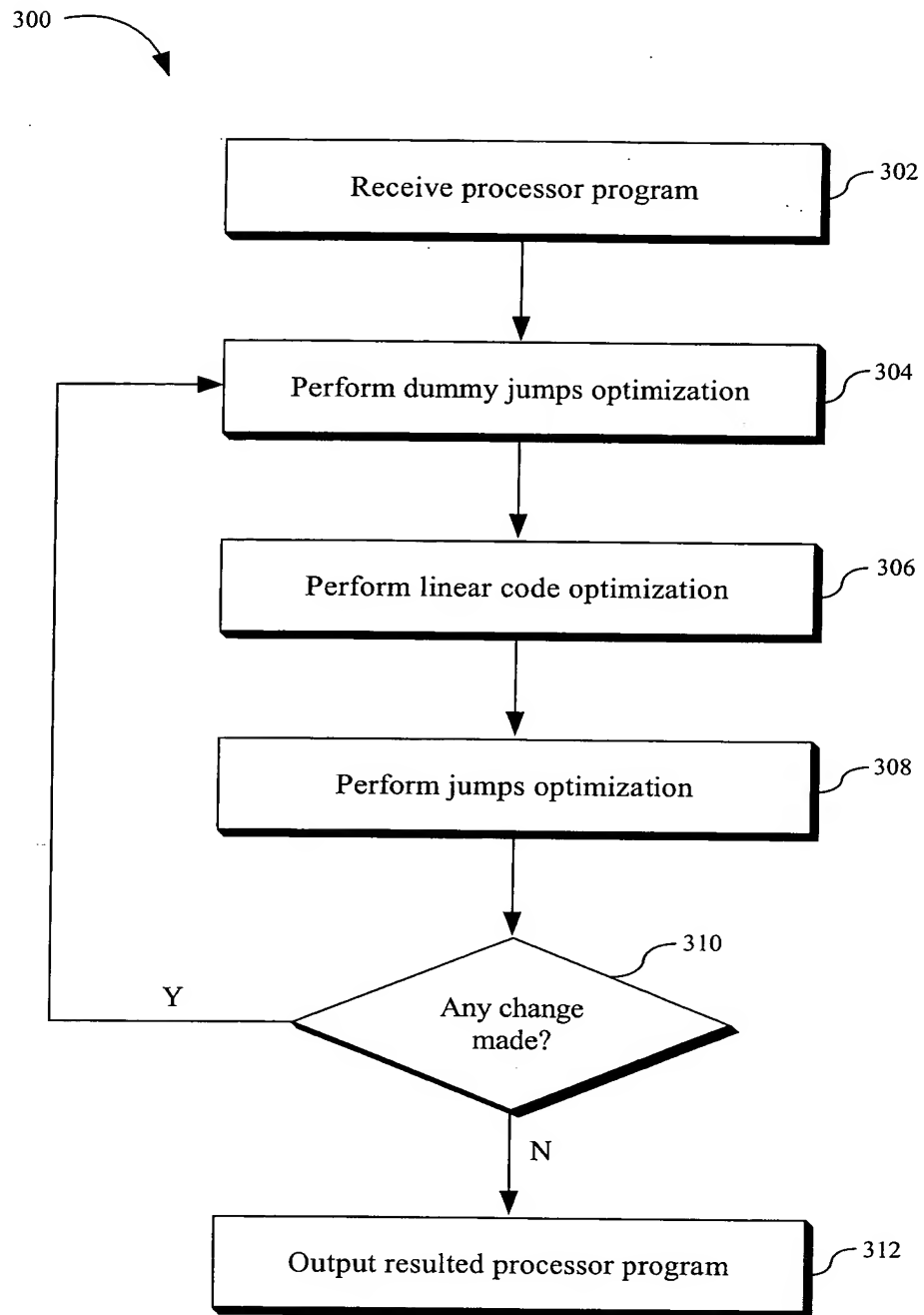
0: READ y exch::pos1;           // function "exch"
1: SET exch::tmp y;             // tmp = y[pos1];
2: READ y exch::pos2;
3: WRITE y y pos1;              // y[pos1] = y[pos2];
4: WRITE y exch::tmp pos2;      // y[pos2] = tmp;
5: RETURN;                      // return from function "exch"

6: SET ALD_0 dir; SET_CONST ALD_1 0; SET_CONST ALD_OP "=";
                                // function "need_exch"
7: ZERO_JUMP ALD_Z 13;         // if (dir==0)
8: READ y need_exch::pos1;
9: PUT y;
10: READ y need_exch::pos2;
11: SET ALD_0 STCK_0; SET ALD_1 y; SET_CONST ALD_OP "<"; DROP 1;
12: PUT ALD_Z; RETURN;          // return (y[pos1]<y[pos2])
13: READ y need_exch::pos1;
14: PUT y;
15: READ y need_exch::pos2;
16: SET ALD_0 STCK_0; SET ALD_1 y; SET_CONST ALD_OP ">"; DROP 1;
17: PUT ALD_Z; RETURN;          // return (y[pos1]>y[pos2])

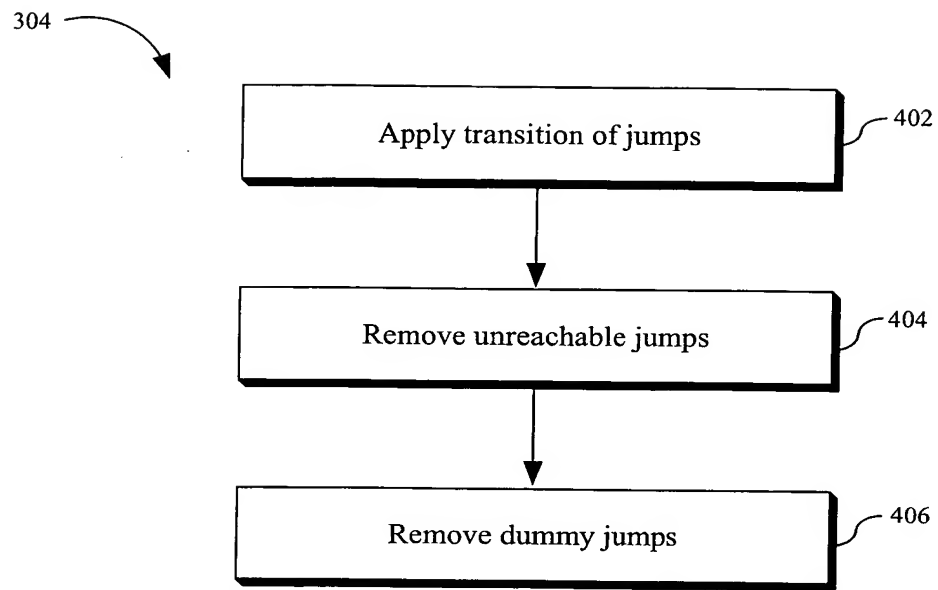
18: SET_CONST i 0;              // main function
19: READ x i;
20: SET y x i;                  // y[i]=x[i]
21: LOOP_INC_NOMORE i 8 19;     // cycle for(i=0;i<10;i=i+1)
22: SET_CONST i 9;              // i=9
23: SET_CONST j 0;              // j=0
24: SET ALD_0 j; SET ALD_1 i; SET_CONST ALD_OP "<";
25: ZERO_JUMP ALD_Z 37;
26: SET ALD_0 j; SET_CONST ALD_1 1; SET_CONST ALD_OP "+";
27: SET k ALD_Z;                // k=j+1
28: SET need_exch::pos1 j;
29: SET need_exch::pos2 k;
30: CALL 6;                     // call function need_exch(j, k)
31: DROP 1; ZERO_JUMP STCK_0 35;
32: SET exch::pos1 j;
33: SET exch::pos2 k;
34: CALL 0;                     // call function exch(j, k)
35: SET j k;
36: JUMP 24;                    // cycle for(j=0;j<i;j=k)
37: LOOP_DEC_NOLESS i 1 23;     // cycle for(i=9;i>=0;i=i-1)
38: FIN;

```

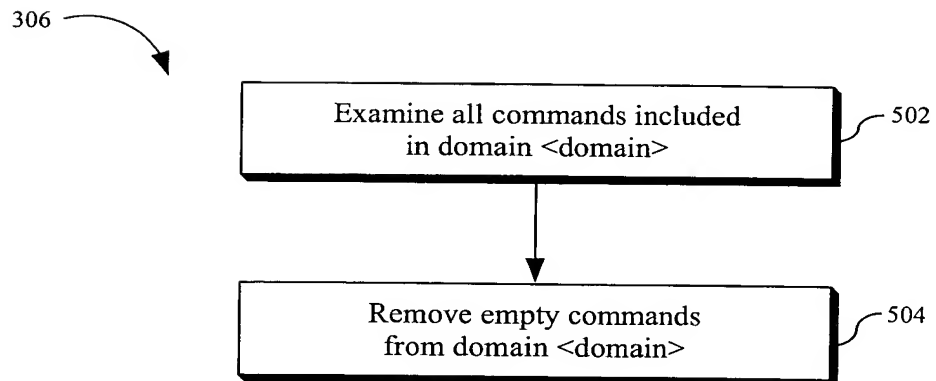
**FIG. 2**



**FIG. 3**



**FIG. 4**



**FIG. 5**

```

0: READ y exch::pos1;           // function "exch"
1: SET exch::tmp y; READ y exch::pos2;
2: WRITE y y pos1;
3: WRITE y exch::tmp pos2; RETURN;

4: SET ALD_0 dir; SET_CONST ALD_1 0; SET_CONST ALD_OP "=";
                                     // function "need_exch"
5: ZERO_JUMP ALD_Z 9; READ y need_exch::pos1;
SET_CONST ALD_OP "<";
6: PUT y; READ y need_exch::pos2;
7: SET ALD_0 STCK_0; SET ALD_1 y; DROP 1;
8: PUT ALD_Z; RETURN;           // return (y[pos1]<y[pos2])
9: READ y need_exch::pos1; SET_CONST ALD_OP ">";
10: PUT y; READ y need_exch::pos2;
11: SET ALD_0 STCK_0; SET ALD_1 y; DROP 1;
12: PUT ALD_Z; RETURN;         // return (y[pos1]>y[pos2])

13: SET_CONST i 0;           // main function
14: READ x i;
15: SET y x i; LOOP_INC_NOMORE i 8 14; // cycle for(i=0;i<10;i=i+1)
16: SET_CONST i 9;           // i=9
17: SET_CONST j 0;           // j=0
18: SET ALD_0 j; SET ALD_1 i; SET_CONST ALD_OP "<";
19: ZERO_JUMP ALD_Z 24; SET ALD_0 j; SET_CONST ALD_1 1;
SET_CONST ALD_OP "+";
20: SET k ALD_Z;           // k=j+1
21: SET need_exch::pos1 j; SET need_exch::pos2 k; SET ALD_0 dir;
SET_CONST ALD_1 0; SET_CONST ALD_OP "="; CALL 5;
                                     // call the second command of the function need_exch(j, k)
22: DROP 1; ZERO_JUMP STCK_0 23; SET exch::pos1 j; SET exch::pos2 k;
CALL 0;           // call function exch(j, k)
23: SET j k; JUMP 18;       // cycle for(j=0;j<i;j=k)
24: LOOP_DEC_NOLESS i 1 17; FIN; // cycle for(i=9;i>=0;i=i-1)

```

**FIG. 6**